A120
KS-Functions

User Manual
DOK-279375.20-0399

Translation of the German Description
DOK-271974.20-0791

Notes

Application Note

Caution The relevant regulations must be observed for control applications involving safety requirements.
For reasons of safety and to ensure compliance with documented system data, repairs to components should be performed only by the manufacturer.

Data, Illustrations, Alterations
Data and illustrations are not binding. We reserve the right to alter products in line with our policy of continuous product development. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us using the form on one of the last pages of this publication.

Training
Schneider Automation offers suitable further training on the system.

Addresses
See addresses for the Technical Support Centers at the end of this publication.

Trademarks
All terms used in this publication to denote Schneider Automation products are trademarks of Schneider Automation.

All other terms used in this publication to denote products may be registered trademarks and/or trademarks of the corresponding Corporations.
Microsoft and MS-DOS are registered trademarks of Microsoft Corporation, Windows is a brandname of Microsoft Corporation in the USA and other countries.
IBM is a registered trademark of International Business Machines Corporation.
Intel is a registered trademark of the Intel Corporation.

Copyright
All rights are reserved. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including copying, processing or by online file transfer, without permission in writing by Schneider Automation. You are not authorized to translate this document into any other language.

© 1999 Schneider Automation GmbH. All rights reserved

Terminology

Note This symbol emphasizes very important facts.

Caution This symbol refers to frequently appearing error sources.

Warning This symbol points to sources of danger that may cause financial and health damages or may have other aggravating consequences.

Expert This symbol is used when a more detailed information is given, which is intended exclusively for experts (special training required). Skipping this information does not interfere with understanding the publication and does not restrict standard application of the product.

Path This symbol identifies the use of paths in software menus.

Figures are given in the spelling corresponding to international practice and approved by SI (Système International d‘ Unités).
I.e. a space between the thousands and the usage of a decimal point (e.g.: 12 345.67).

Objectives

This documentation describes a special product feature of the programmable controller Modicon A120: the Communication Interface-functions (CI-functions). The CI-functions are a special feature of the Modicon A120 arithmetic logical units (ALU's), which allow the connection of external devices to the programming interface (RS232, all ALU's) or the communication interface (Modnet 1-port, only ALU202) of the Modicon A120.

The following external devices can connected:

- Operating and monitoring devices
- Portable hand-held devices
- Permanently installed parametrization modules
- All kinds of operator panels
- Personal Computer (production data aquisition, PDA; visual display systems)
- Other intelligent peripheral devices.

The documentation on hand gives all information necessary in order to be able to use the CI-function for individual applications. The main emphasis is the coupling of external devices to the A120-programming port (RS 232 C).

Arrangement of This Guide

Chapter 1 This chapter describes the idea of the CI-functions and the possible solutions on basis of the CI-functions.

Chapter 2 This chapter explains the message interchange, the structure of the messages and the message content: the CI-functions.

Chapter 3 This chapter explains the data exchange between the programmable controller A120 and the periperal, in sequence charts.

Chapter 4 In this chapter some examples about the CI-functions are shown.

Validity Note

The functions explained in this documentation are applied to the system software A120. Validity march 1991.

Supplement to User Manual A120 KS Functions, DOK-279375.20-0399

KS Functions for Modicon Micro

The KS functions described in the following manual are valid without any restrictions also for Modicon Micro.

KS Functions for Modicon A250 and the A120 central processing units ALU 204/ ALU 205

The KS functions described in the following manual are available from AKF125 version 3.0 also for Modicon A250. The KS functions are available also for the A120 central processing units ALU 204 and ALU 205.

In case of these applications the following deviations in comparison to the application with A120 should be considered (for this compare user manual page 13, Figure 9):

- The following A120 system messages do not exist with A250 and ALU204/5:
  - write system definition list
  - write target equipment list
  - start user program in cold restart mode
  - start user program in hot restart mode

☞ **Note** Modicon A250 and the CPUs ALU204/5 are basically started with the use of hardware jumpers.

- With the following system messages, for the A250 depending on the system a different data format must be used:
  - read system status list
  - read system definition list

- Since A250 possesses considerably more interfaces than A120, the declared numbers cannot be accessed freely. This means the user must get a free declaration number from A250 and cannot select it freely (see user manual page 31, chapter 4.4.1.1).

- From AKF125 version 4.0 it is possible to read and write floating point values.

- As with Modicon A120 and Modicon Micro, we recommend to use the ”single read and write on markers” as the standard method (see user manual page 28, chapter 4.3).

- As with Modicon A120 and Modicon Micro it is meaningful to store markers of the same type in a marker chain and to access these as a block (see user manual page 4, Table 1).

Table of Contents

Chapter 1 The KS-Functions 1

1.1 Introduction 2
1.2 Possible Peripherals 2
1.3 Additional Possibilities using the KS-Functions 4

Chapter 2 Display of Message Structure and Timing 5

2.1 Message Interchange 6
2.1.1 Polling Operation 6
2.1.2 The Master-Slave-Communication with Acknowledgement 6
2.1.3 The Master-Slave-Communication with Information Response 7
2.1.4 The MasterSlaveCommunication with Error Message 7
2.1.5 Message Types 8
2.1.6 The Structure of the Message Elements 8
2.2 The Structure of the System Message 9
2.2.1 The Message Header 9
2.2.2 The Message Content 10
2.2.3 Message Protection 14
2.3 The Communication Time Frame 16

Chapter 3 Principle Program Structures 19

3.1 How to Keep the Communication Active 20
3.2 How to Create a System Message 21
3.3 Analyses of Response Message 23

Chapter 4 Examples 25

4.1 General 26
4.2 Starting and Stopping of the PC-User Program 27
4.2.1 Starting of the System 27
4.2.2 Stopping of the System 27
4.3 Single Read of three Successive Marker Bits 28
4.4 Multiple Access to declared Data 29
4.4.1 Structure of a Declaration List 29
4.4.1.1 How to read a free declaration number 29
4.4.1.2 Declaration of four successive marker bytes for multiple read 29
4.4.1.3 Terminate Data Declaration resp. declare Send Condition 30
4.4.2 Execution of a Declaration List 32
4.4.2.1 Periodic Scanning of a Declaration List 32
4.4.2.2 Polling while waiting for the Send Bit 32
4.4.3 Clearing of Declarations 33
4.5 Using Timers and Counters 34

Figures

Figure 1 Possible Peripherals . . . 3
Figure 2 Polling Operation . . . 6
Figure 3 Master-Slave-Communication with Acknowledgement . . . 6
Figure 4 The Master-Slave-Communication with Information Response . . . 7
Figure 5 The Master-Slave-Communication with Error Message . . . 7
Figure 6 Structure of the System Message . . . 9
Figure 7 Structure of the Message Header . . . 10
Figure 8 The Message Content of the System Message for the Request of the Master . . . 11
Figure 9 The Message Content of a System Message for a Slave Response . . . 13
Figure 10 Generation of the Protection Byte ”S” . . . 15
Figure 11 Timing Diagram Polling Mode . . . 16
Figure 12 System Message Slave . . . 17
Figure 13 Response Message Slave Master . . . 18
Figure 14 Communication Operation . . . 20
Figure 15 Creation of System Messages . . . 22
Figure 16 Evaluation of the Response Message . . . 23
Figure 17 System Message: Start of PC-User Program . . . 27
Figure 18 Response Message of the Modicon A120 . . . 27
Figure 19 System Message: Stop PC-User Program . . . 27
Figure 20 Response Message Modicon A120 . . . 28
Figure 21 System Message for Single Read of Signal-Memory-Data . . . 28
Figure 22 Response Message of the Modicon A120 . . . 28
Figure 23 System Message: Read the smallest free Declaration Number . . . 29
Figure 24 Response Message of the Modicon A120 . . . 29
Figure 25 Declaration of four successive marker bytes . . . 30
Figure 26 Response Message of the Modicon A120 . . . 30
Figure 27 End Data Declaration . . . 30
Figure 28 Declare Send Condition . . . 31
Figure 29 Declare Send Condition . . . 31
Figure 30 Response Message of the Modicon A120 . . . 31
Figure 31 Send Data of the Declaration List 27 . . . 32
Figure 32 Response Message of the Modicon A120 . . . 32
Figure 33 Cleanse single Declaration List in A120 . . . 33
Figure 34 Clear all Declaration Lists in A120 . . . 33

Tables

Table 1 Operand Overview . . . 4

Chapter 1
The KS-Functions

This chapter describes the idea of the KS-functions and possible solutions or applications.

1.1 Introduction

Mini programmable controllers (PCs) – like the Modicon A120 – are used in substantial quantities for machine control purposes and for the control of small applications in the process industry.

Besides the display of message texts and alarms it must be possible to modify setpoint values and other parameters in the programmable controller. Very often color graphic displays of the process are required, process data have to be scanned and stored resp. processed on a Personal Computer for production data acquisition (PDA). In some cases special sensors – like bar-code readers – are connected.

To be able to fulfill these demands, the openness of a programmable controller becomes more and more important. The programmable controller Modicon A120 is therefore designed as an open controller, the KS-functions are part of this concept.

This documentation shall give the necessary background information to the user in order to make this specific product feature easier to understand.

1.2 Possible Peripherals

The name KS-functions defines a feature of the Modicon A120 arithmetic logical controllers (ALUs), which allows the connection of external devices to the programming or the communication interface of the ALUs.

The following peripherals are possible:

- Operating- and monitoring devices
- Portable handheld-devices
- Permanently installed parametrization modules
- All kinds of operator panels
- Personal Computers (production data acquisition, PDA; visual display system)
- Other intelligent peripheral devices.

This documentation includes all information necessary to be able to use the KS-functions for individual applications. The main emphasis in this manual is the connection of devices to the Modicon A120 programming interface (RS 232 C), i.e. as a real point-to-point connection. In principle it is also possible to use the KS-functions via the communication interface (ALU 202). An example for this is the MMI-product Viewstar 200XA for A120.

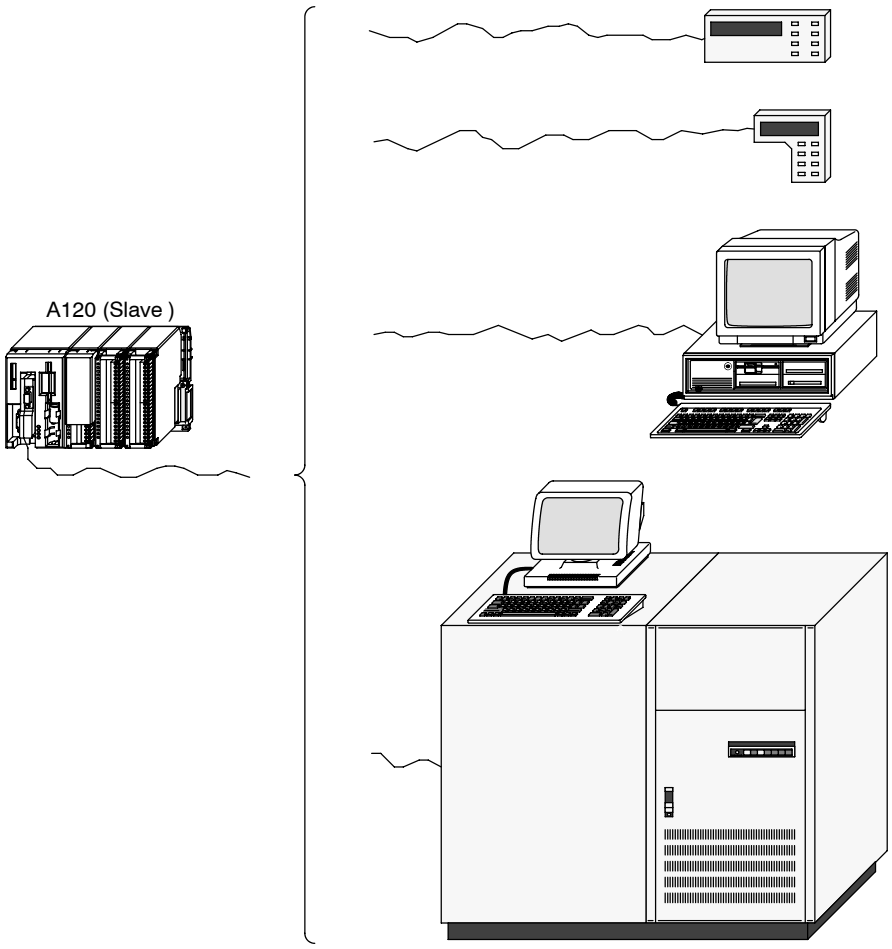


Figure 1 Possible Peripherals

1.3 Additional Possibilities using the KS-Functions

Using the KS-functions the peripherals can be used for the following:

- control of the programmable controller
- define data blocks for cyclical read or write functions
- read data once or cyclical
- write data once or cyclical
- read data event-controlled, i.e. receiving data from the programmable controller without request.

All operand types with different system-defined address ranges can be used when reading or writing data. In Figure 2 please find the possible address ranges. The size of the resp. address range is defined in the AKF12EN-user program.

Table 1 Operand Overview

Abbreviation	Meaning	Operands (Addresses)				
I	Inputs binary	I	1.1	. . .	I	18.16
IB	Byte Inputs	IB	1.1	. . .	IB	18.16
IW	Word Inputs	IW	1.1	. . .	IW	18.16
ID	Double Word Input	ID	1.1	. . .	ID	18.16
Q	Binary Outputs	Q	1.1	. . .	Q	18.16
QB	Byte Output	QB	1.1	. . .	QB	18.16
QW	Word Outputs	QW	1.1	. . .	QW	18.16
QD	Double Word Outputs	QD	1.1	. . .	QD	18.16
M	Marker Bit	M	1.1	. . .	M	125.32
MB	Marker Byte	MB	1	. . .	MB	3970
MW	Marker Word	MW	1	. . .	MW	1985
MD	Marker Double Word	MD	1	. . .	MD	992
SM	System Marker Bit	SM	1	. . .	SM	50
SM	System Marker Bit (I/Os)	SM	1.1	. . .	SM	18.1
SMB	System Marker Byte	SMB	1	. . .	SMB	20
SMB	System Marker Byte (analog I/Os)	SMB	1.1	. . .	SMB	18.1
SMW	System Marker Word	SMW	1	. . .	SMW	10
SMD	System Marker Double Word	SMD	1	. . .	SMD	5
T	Timer	T	1	. . .	T	567
TIW	Timer Actual Value Word	TIW	1	. . .	TIW	567
TSW	Timer Set Value Word	TSW	1	. . .	TSW	567
C	Counter	C	1	. . .	C	794
CAW	Counter Actual Value Word	CAW	1	. . .	CAW	794
CSW	Counter Set Value Word	CSW	1	. . .	CSW	794

Chapter 2
Display of Message Structure and Timing

This chapter explains the message interchange, the structure of the messages and the message content: the KS-functions.

2.1 Message Interchange

The communication between the programmable controller (PC) Modicon A120 and the peripheral device is based on the master-slave-principle. The PC is always slave station. The peripheral therefore must be able to keep the data exchange active with the programmable controller Modicon A120.

The following types of data exchange are possible:

2.1.1 Polling Operation

The master station (Personal Computer, ...) scans the slave station in regular time intervals, in order to control the transmission link and to synchronize the exchange of data. This is done with short messages (SMG), which contain only a systemdefined HEX-character but no information.

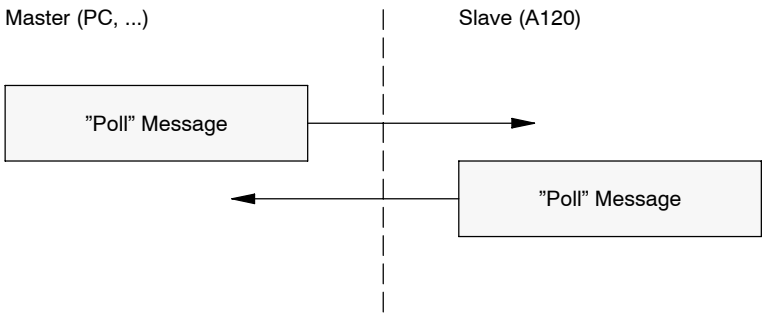


Figure 2 Polling Operation

2.1.2 The Master-Slave-Communication with Acknowledgement

The masterstation sends a message with control information or data to the slave (request). The slave then confirms the message with a long message similar to the structure of the master message (response). Due to the fact that response messages normally cannot be sent immediately, therefore short messages (SMG), which have a similar structure as the polling messages, are sent in the meantime.

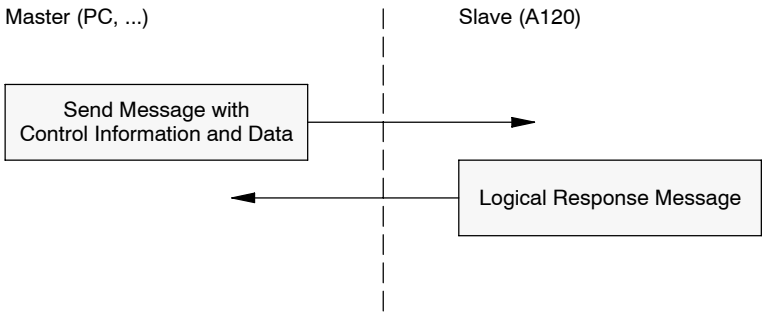


Figure 3 Master-Slave-Communication with Acknowledgement

2.1.3 The Master-Slave-Communication with Information Response

The master sends a message with a data request to the slave (request). The slave answers with a response message which is similar to the master message (response) but including the requested information. Due to the fact that the response message is not sent immediately, therefore short messages, which are similar structured like polling messages, are sent in the meantime.

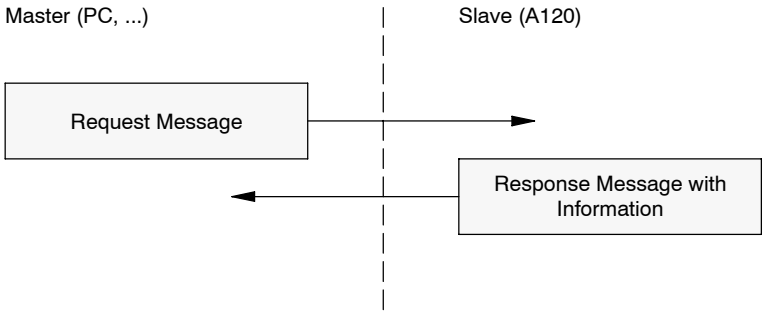


Figure 4 The Master-Slave-Communication with Information Response

2.1.4 The MasterSlaveCommunication with Error Message

The master sends a message with a control information or a data request to the slave (request). In case that an error occurs, e.g. data fields are read which have not been configured, then the slave responds with an error message similar to the master message (response). This error message contains only the first four bits, with a changed status. The status is different to zero! For details please see Figure 10.

Due to the fact that the respective response message cannot be transmitted immediately after the request, therefore short messages are interchanged in the meantime.

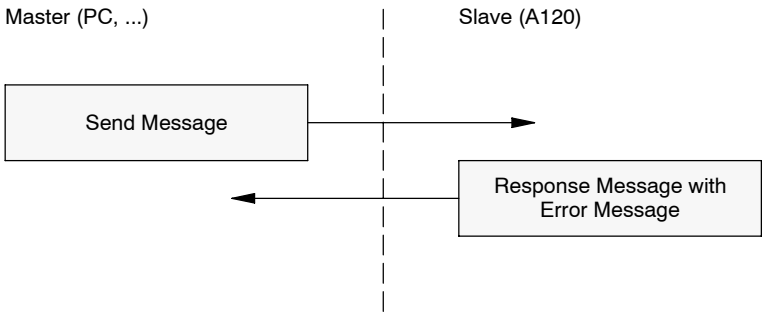


Figure 5 The Master-Slave-Communication with Error Message

2.1.5 Message Types

Two different message types - the short message and the system message - are possible for the four communication principles mentioned. The structure of the system messages is explained in details in chapter 2.2.

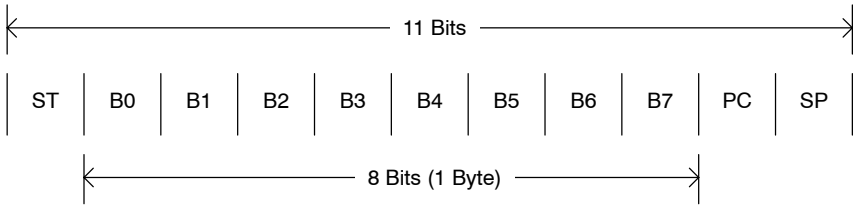
The short messages only consist of a short message header. They are used for the polling operation as well as for the normal communication. In accordance with the definition the following hexadecimal numbers are used for the polling operation and for the communication:

- Short message (SMG) for the communication ”80”
- Short messages (SMG) for the polling operation ”81 ”

The short message (SMG80) is used exclusively for the synchronization during the communication and is transmitted between the different system messages, i.e. while the master expects a system message from the slave. This is also valid for the send bit declaration. For the polling operation ST 81 is used exclusively.

2.1.6 The Structure of the Message Elements

Each message consists of different message elements. The short message has only one message element, whereas the system message can have up to 69 message elements.



Explanation:

- ST = Start Bit
- B0 up to B7 = Data Bits
- PC = Parity Bit
  (Vertical Parity = odd number of bits logical ”1”)
- SP = Stop Bit

The length of a message element is 11 bits when using the A120 programming interface.

2.2 The Structure of the System Message

The system message is subdivided into three element groups:

- Message Header
- Message Content
- Message Protection

1	Message Part	Message Header	Message Content	Message Protection
2	Bytes	from A up to D5	from D6 up to D64	S

Figure 6 Structure of the System Message

The system message is merged using the before mentioned parts.

☞ **Note** All values transferred into the message have to be in hexadecimal format.

2.2.1 The Message Header

The message header consists of 9 bytes. Only the first three bytes ”A”, ”F” and ”F1” are of importance to the user. The hexadecimal value of the byte ”F2” is always ”F0”. The header comprises the following information:

- The direction of the message
- The length of the message content
- The type of message

The structure of the message header is explained in the following.

1	Embedded Command	A	F	F1	F2	D1	D2	D3	D4	D5
2	Message Header for all System Message	0	R	L	F0	0	0	1	0	0
3	Message Header for Short Message in Polling Operation	81	N	N	N	N	N	N	N	N
4	Message Header for Short Message in Communication Operation	80	N	N	N	N	N	N	N	N

Values to be Entered / Range of Values	Range of Values	Legend
Message Direction		
From Master to Slave Modicon A120 (Request)	0C	R
From Slave Modicon A120 to Master (Response)	0D	R
Length of the Data Area D1..D64		
Number of Bytes within Message	(0...5) 6..64	L

Figure 7 Structure of the Message Header

The message type is coded in byte ”A”. A ”0” stands for a system message. The short messages (length only one byte) are mentioned only in order to complete the picture.

The byte ”F” contains information whether a message is transmitted from the A120 or to the A120. Byte ”F1” defines the number of data bytes D. This value can be entered only after the definition of the message content.

2.2.2 The Message Content

The different KS-functions, operand types and operand contents are defined in the message content from the master or the resp. slave station. In this context the message direction is of importance for the assignment of the different functions to the transmitted bytes. Therefore the send message (master → slave) and the receive message (slave → master) should be written separately.

For both senddirections separate layout plans have been created in order to simplify the composition of the message contents for the different functions. The hatched sections in the plan show the message elements where the user has to fill-in values, dependent on the communication task. Bytes which are marked with ”N” can have any value. It is recommended to set these bytes to ”0”. Figure 8 shows the structure of the message content in ”master → slave” direction.

Operand Description		SYS		Representation
Marker Bit	(M)	20		1 Byte / Operand
Marker Byte	(MB)	21		1 Byte / Operand
Marker Word	(MW)	22		2 Byte / Operand
Marker Double Word	(MD)	23		4 Byte / Operand
System Marker Bit	(SM)	28		1 Byte / Operand
System Marker Byte	(SMB)	29		1 Byte / Operand
System Marker Word	(SMW)	2A		2 Byte / Operand
Sytsem Marker Double Word	(SDW)	2B		4 Byte / Operand
Timer Bit	(T)	* 30	*1 40	1 Byte / Operand
Timer Actual Value	(TAW)	* 2C	*1 3C	2 Byte / Operand
Timer Setpoint Value	(TSW)	* 2D	*1 3D	4 Byte / Operand
Counter Bit	(C)	* 31	*1 41	1 Byte / Operand
Counter Actual Value	(CAW)	* 2E	*1 3E	2 Byte / Operand
Counter Setpoint Value	(CSW)	* 2F	*1 3F	4 Byte / Operand

*** Version 2/3 of the A120 firmware**
***1 Version 3**

Operand Description		I/O	Representation
Input Bit	(I)	0	1 Byte / Operand
Input Byte	(IB)	1	1 Byte / Operand
Input Word	(IW)	2	2 Byte / Operand
Input Double Word	(ID)	3	4 Byte / Operand
Output Bit	(Q)	10	1 Byte / Operand
Output Byte	(QB)	11	1 Byte / Operand
Output Word	(QW)	12	2 Byte / Operand
Output Double Word	(QD)	13	4 Byte / Operand

Embedded Command	D6	D7	D8	D9	D10	D11	D12	D13	D14	...	...	D64
Start User Program (UP) acc. to Hardware Settings	1	1	N	N								
Start UP in Cold Restart Mode	1	2	N	N								
Start UP in Hot Restart Mode	1	3	N	N								
Stop UP	1	4	N	N								
Standardize Signal Memory (SM) System specific	1	5	N	0								
Standardize Signal Memory all Ranges	1	5	N	255								
Delete Declaration Number	1	7	RoV	N								
Terminate Data Declaration	1	9	RoV	N								
Declare SM-Data (multiple read)	2	1	RoV	0	SYS	NB	AD	AD				
Declare SM-Data (multiple read)	2	1	RoV	1 18	IO	NB	AD	AD				
Declare SM-Data (multiple write)	2	6	RoV	0	SYS	NB	AD	AD				
Declare SM-Data (multiple write)	2	6	RoV	1 18	IO	NB	AD	AD				
Single Read SM-Data	5	1	N	0	SYS	NB	AD	AD				
Single Read SM-Data	5	1	N	1 18	IO	NB	AD	AD				
Read System Status List	6	N	1	N								
Read System Definition List	6	N	2	N								
Read free Declaration Number	6	N	3	N								
Multiple Data Read	6	N	Rov	N								
Single Write SM-Data	7	1	N	0	SYS	NB	AD	AD	D	**D**	D	D
Single Write SM-Data	7	1	N	1 18	IO	NB	AD	AD	D	**D**	D	D
Write System Definition List	8	N	6	N								
Write Target Equipment List	8	N	7	N								
Write Time and Date	8	N	8	N	D	D	D	D	D	D	D	D
Multiple Data Write	8	N	RoV	N	D	D	D	D	D	D	D	D

Values to be entered / Range of Values (RoV)	Range of Values	Legend
Declaration Number		
Range of Values ALU 200	17 ... 37	RoV
Range of Values ALU 201	17 ... 127	RoV
Range of Values ALU 202	17 ... 127	RoV
Number (NB):		
Value for Number of used		
Operands SYS or IO	12 ... 51	NB
Start Address:		
Lowest Address of the afore mentioned		
Number of Operands		AD
Range of Data:		
Information to be transmitted		D
Irrelevant Range of Data:		
Any values can be entered		N
Variables:		
Values which have to be defined from the user		
for each transmission of data		

Attention: at data byte D9 decimal values are shown

Figure 8 The Message Content of the System Message for the Request of the Master

Definite values can be assigned to the bytes D6 (function byte) and D7 (subfunction byte). For the data byte D8 (declaration byte) at this position the resp. declaration number has to be entered for functions which operate with declared data. The range of values (RoV) for the declaration number depends on the selected ALU-type and must not be exceeded. Via the content of byte D9 (I/O slot number) a module of the A120 can be addressed (e.g. I/O module). Example: To be able to read out the input conditions of an I/O module in slot 3, the slot number ”3” has to be entered.

In byte D10 the resp. operand type in hexadecimal format must be entered if necessary. Byte D11 describes the number of operands per transmitted message. In byte D12 and D13 the start address is entered (first address of the operand string). Example: The markers ”999” up to ”1001” have to be transmitted. In byte D11 the value 3 (number of markers) and in bytesD12 up to D13 the value ”999” (in hexadecimal format) has to be entered.

In the byte range D14 up to D64 the send station of the message can store the operand values which have been defined in D10 up to D13. The different operand types require one up to four bytes for the operand value. Example: A marker bit (1 bit) requires one byte, a marker double word (32 bit) requires four bytes in total.

The black arrow in Figure 8 marks the number of bytes necessary for the resp. function which have to be reserved for the message.

Meaning of the Error Code S	Status Value
No Error	00
Wrong CI Function Group Number	01
Wrong Subfunction Number	02
Declaration Number too big	03
Wrong Station Number	04
Wrong Operand Type	05
Range exceeded	06
Alloction Error	07
Non-Permissible Message Sequence	08
Response Length Error	09
Declaration Length Error	0A
No free Declaration Numbers available	0B

Embedded Command	D6	D7	D8	D9	D10	D11	D12	D13	D14	...	...	D64
Start user program (UP) according to HW Settings	1	1	N	S								
Start UP in Cold Restart Mode	1	2	N	S								
Start UP in Hot Restart Mode	1	3	N	S								
Stop UP	1	4	N	S								
Standardize Signal Memory (SM) System specific	1	5	N	S								
Standardize Signal Memory all Ranges	1	5	N	S								
Delete Declaration	1	7	RoV	S								
Terminate Data Declaration	1	9	RoV	S								
Declare SM-Data (multiple read)	2	1	RoV	S								
Declare SM-Data (multiple read)	2	1	RoV	S								
Declare SM-Data (multiple write)	2	6	RoV	S								
Declare SM-Data (multiple write)	2	6	RoV	S								
Single Read SM-Data SYS	5	1	N	S	D	D	D	D	D	D	D	D
Single Read SM-Data IO	5	1	N	S	D	D	D	D	D	D	D	D
Read System Status List	6	N	1	S	PS	PS	PS	PS	PS	PS	PS	PS
Read System Definition List	6	N	2	S	PS	PS	PS	PS	PS	PS	PS	PS
Read free Declaration Number	6	N	3	S	D							
Multiple Data Read	6	N	RoV	S	D	D	D	D	D	D	D	D
Single Write SM-Data	7	1	N	S								
Single Write SM-Data	7	1	N	S								
Write System Definition List	8	N	6	S								
Write Target - Equipment List	8	N	7	S								
Write Time and Date	8	N	8	S								
Multiple Data Write	8	N	RoV	S								

Values to be entered / Range of Values	Range of Values	Legend
Declaration Number:		
Range of Values ALU 200	17..37	RoV
Range of Values ALU 201	17..127	RoV
Range of Values ALU 202	17..127	RoV
Range of Data:		
Information to be transmitted		D
Predefined Sequence (PS) of Information:		
Gives predefined information sequences		PS
back to the user with actual data		
Irrelvant Range:		
Any values can be entered		N
Variabels:		
Values which have to be defined from the user		
for each transmission		

Figure 9 The Message Content of a System Message for a Slave Response

The response messages are broken down according to the above mentioned description. Please be aware that the bytes in the response message have partly a different meaning.

The meaning of the bytes D6 up to D8 correspond to the meaning of the bytes in a send message.

Byte D9 can be used from the slave station Modicon A120 in order to transmit different error messages to the master station. In case that the A120 station sets D9 to ”0” then no errors are detected and the system message which was send from the master was according to the before mentioned conditions.

In the bytes D10 up to D64 the slave station transmits the values of the before declared operands to the master station.

The data ranges marked with ”PS” (for details please see Figure 9) are predefined in their order. With that it is possible to read out complete internal lists of the programmable controller, e.g. system status lists or system definition lists at once, i.e. in one message. The procedure is this case is the same like when reading out operand blocks which are predefined via declaration numbers, with the difference that the declaration numbers cannot be freely selected but are specified from the system.

2.2.3 Message Protection

The protection of the data transmission is done in two levels. For the physical layer the convention of the RS232 - port secures automatically the control of the data transmission. This level is less important to the user.

For the logical level a so-called protection byte is used in order to ensure a correct data transmission. This byte is added to each system message. For further details please see Figure 10.

Embedded Command	S
Value for the Protection Byte	PC

Message Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Byte A	0	0	0	0	0	1	1	1
Byte F	0	0	1	0	1	0	1	1
Byte F1	1	1	1	1	0	0	0	0
Byte F2	0	0	1	1	1	1	1	1
Byte D1	1	1	0	0	0	1	1	1
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
Byte D64	0	0	0	0	0	0	0	0
	|	|	|	|	|	|	|	|
Parity Check to ”1”	|	|	|	|	|	|	|	|
	↓	↓	↓	↓	↓	↓	↓	↓
Byte S	1	1	0	1	1	0	1	1

Figure 10 Generation of the Protection Byte ”S”

The data transmission is protected via a odd parity check, i.e. all bits which are necessary for the parity are checked for the number of bits which are logical ”1”. Is the number of these bits an even number, then the corresponding parity bit is set to ”1” (high), in the other case it is set ”0” (low). The parity check is carried out vertically. For this all bytes, max. byte 1 up to byte 64, are listed is binary digits. Each column now represents a bit string with an allocated parity bit. The parity bit of each column is collected in a byte and represents the value of the protection byte. This binary value must now be converted into a hexadecimal format. This value now represents the last value in the message.

2.3 The Communication Time Frame

The developer of peripheral devices has to pay special attention to a correct time synchronization of the messages exchanged between the master and the slave station. In case that the master cannot fulfill the necessary timing code, then unstable conditions or in the worst case a faulty communication can be the result.

The timing code of the programmable controller Modicon A120 is oriented on the requirements of Personal Computers equipped with the MS-DOS-operating system (registered trademark of Microsoft).

The developer has to differ between the polling time (t_P) and the waiting time (t_W). The polling time describes the time interval in which the corresponding communication partner has to send a message in the opposite direction. The waiting time describes the max. time interval after which the slave or the master station consider the communication as faulted and after which the stations generate the corresponding countermeasures. The reference of the resp. time to the station type, in the picture below is carried out with M (master station) or S (slave station).

Please pay special attention to the waiting time t_WS. In case that the predefined timing code for the resp. operating mode is exceeded, then the RS 232 - port is standardized and the I/O buffer in the Modicon A120 is deleted.

The timing code for the polling operation is dimensioned in its operating description to 55 msec. Figure 11 shows the timing diagram for this operating mode.

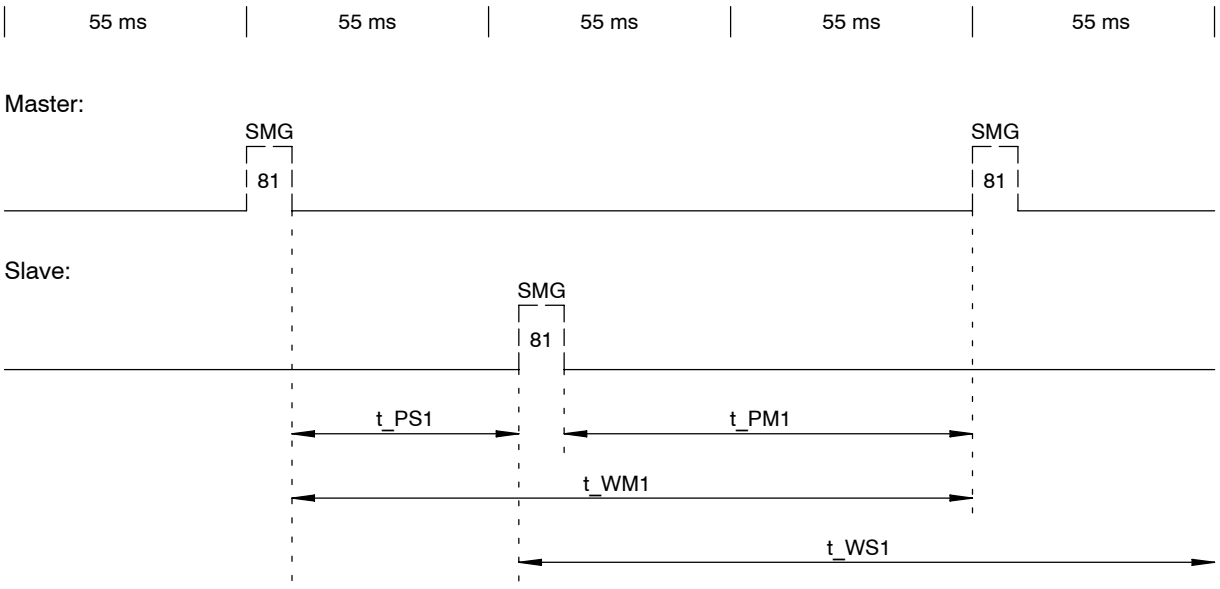


Figure 11 Timing Diagram Polling Mode

The time t_PS is the reaction time of the controller to a short message (SMG) 81 which has been sent from the master. For this reaction of the programmable controller a response time of up to 110 msec. is allowed for the slave. In case that this time is exceeded, then the program in the master station must be able to recognize that as an error (by selecting an appropriate waiting time t_WM) and must repeat the call. Theoretically the master can react directly to the polling signal SMG 81 of the slave station. In order to prevent the master from receiving too many polling signals, the master station has a time interval of max. 120 msec. before it has to send a new polling signal.

When changing from polling mode to the operating mode ”system messages”, the timing code is reduced to 10 msec. Due to the fact that the system message is passing the serial port byte by byte and not complete, the developer has to make sure that the period of time between two bytes does not exceed 40 msec. Figure 12 shows the time structure when sending a message from the master to the slave station.

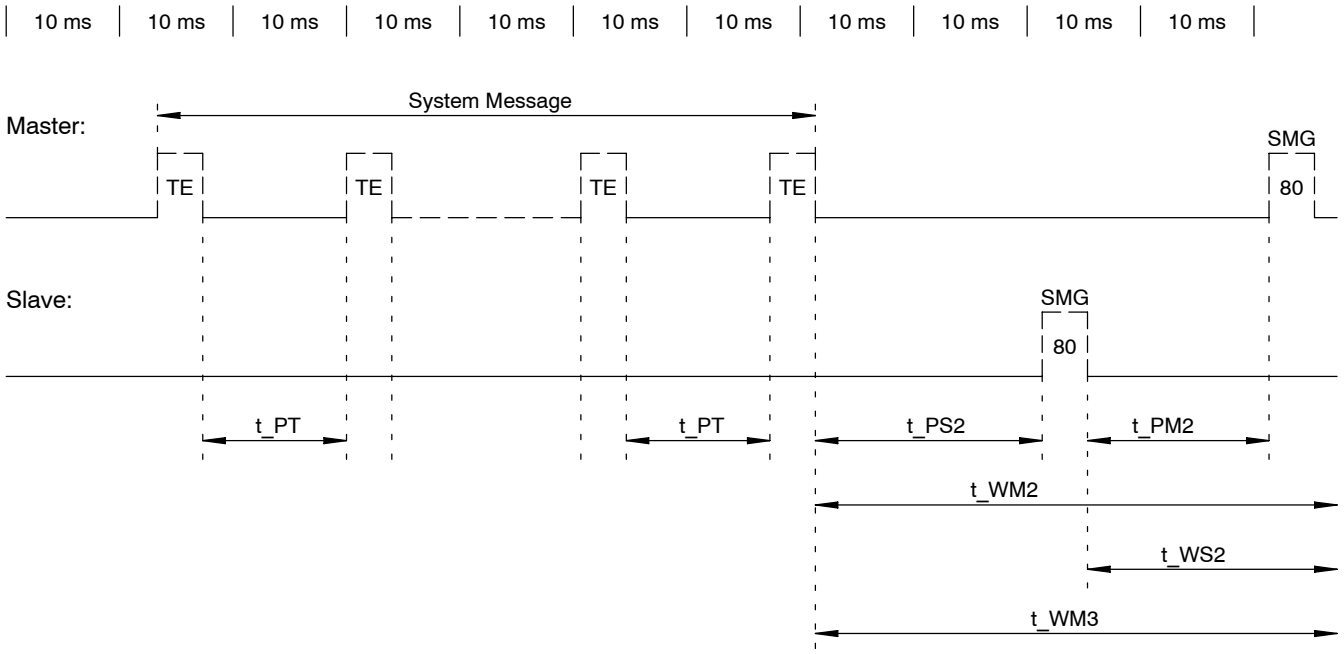


Figure 12 System Message → Slave

The slave station recognizes the end of a system message when the time interval ”t_PT + 20msec. = t_PS” is passed after the last byte to the slave. The slave station sends a short message ”SMG 80” to the master when the message end has been recognized. The master then demands (by sending ”SMG 80” short messages) the slave station to send the logical acknowledgement (after a time interval ”t_PM = t_PT + 0 up to 20msec.”). Correspondingly to the polling operation also here different waiting times have to be taken into consideration. In case that these waiting times are exceeded, then a communication error has to be recognized by the master station.

Figure 13 shows the timing diagram for the send sequence of a response message from the slave to the master station. The time intervals are correspondent to the send operation from master to slave.

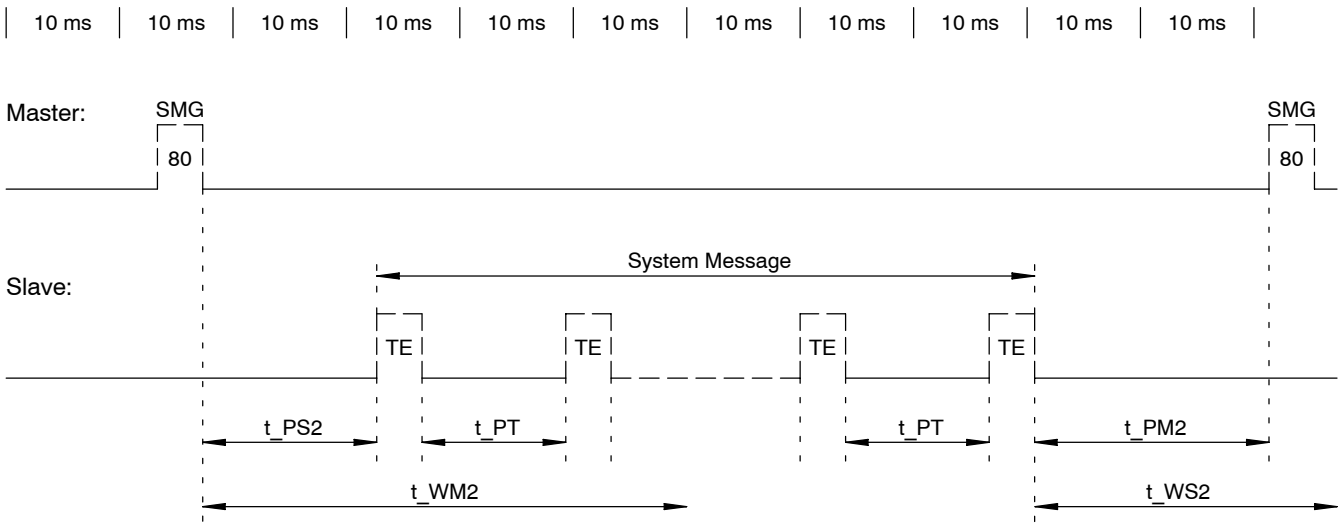


Figure 13 Response Message Slave → Master

Chapter 3
Principle Program Structures

This chapter explains the data exchange between the programmable controller Modicon A120 and the peripheral, in sequence charts.

3.1 How to Keep the Communication Active

The start of the polling routine (loop 1) is the first program element which starts the communication. In case that the master can fulfill the requirements concerning the timing code as mentioned in chapter 2.3, then loop 1 is processed permanently until a system message is sent.

Loop 2 shall keep the polling operation active in case that insignificant irregularities happen. When all three tests fail it has to be guaranteed that the polling operation (loop 3) is activated again.

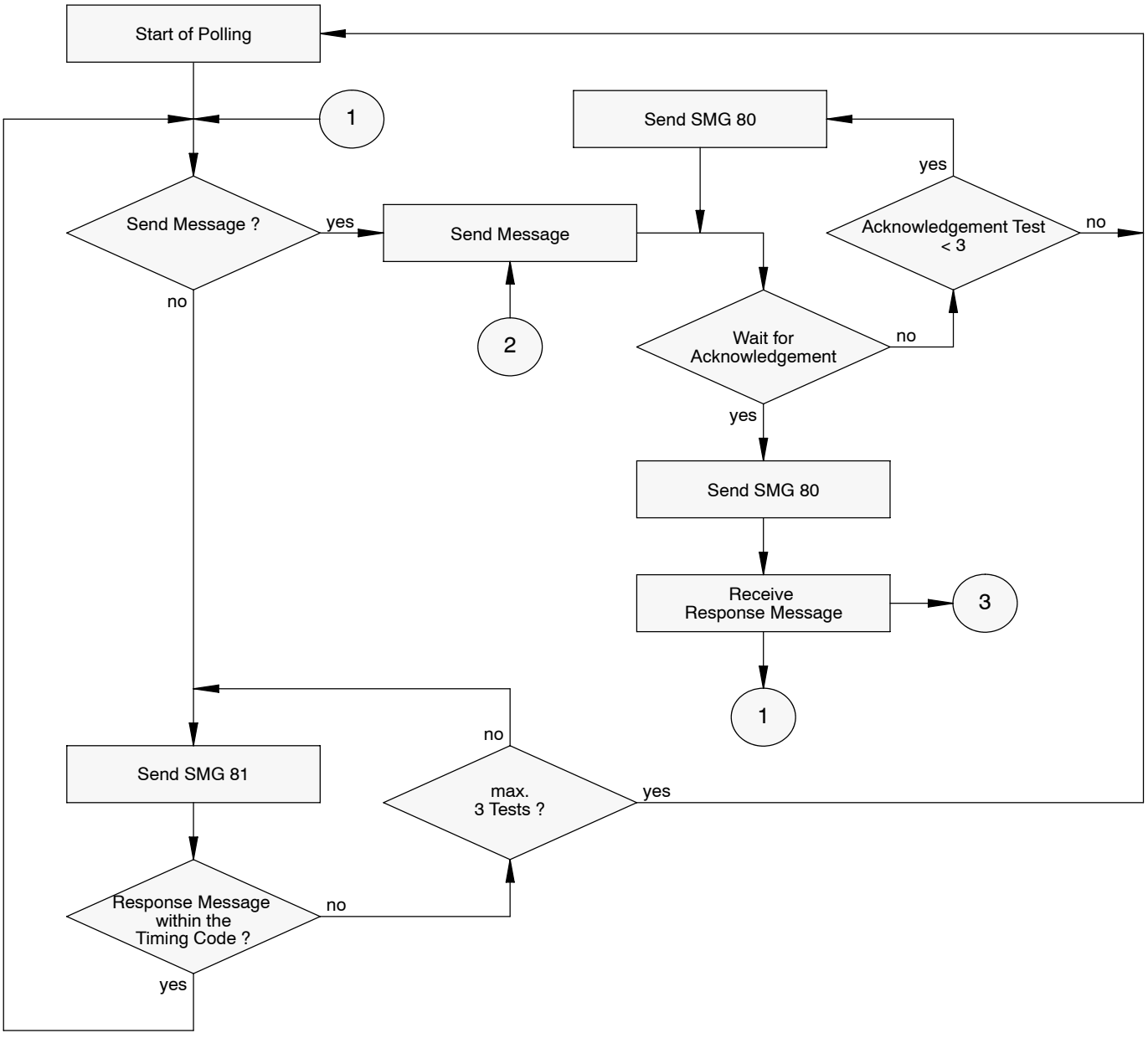


Figure 14 Communication Operation

3.2 How to Create a System Message

The creation of system messages has to be carried out in parallel to the before mentioned communication routine. These program parts my influence the communication routine in the way that an unpermissible increase of the operating time occurs. In case that the masters computing power is too low, then an increased priority in the program sequence has to be assigned to communication routine. Figure 15 shows the recommended steps when creating a system message. The upper part of the chart shows decisions related to the selected function, mode, number and content of different operands, while the second part contains standard tasks, which bring the message in its final and correct form.

The developer of the program can define how the necessary decisions for the first part are made, i.e. via menu methods or programs which support special applications.

How the the complete message is linked into the communication routine is indicated with position 2. In this process the communication routine guarantees trouble-free transmission within the timing code.

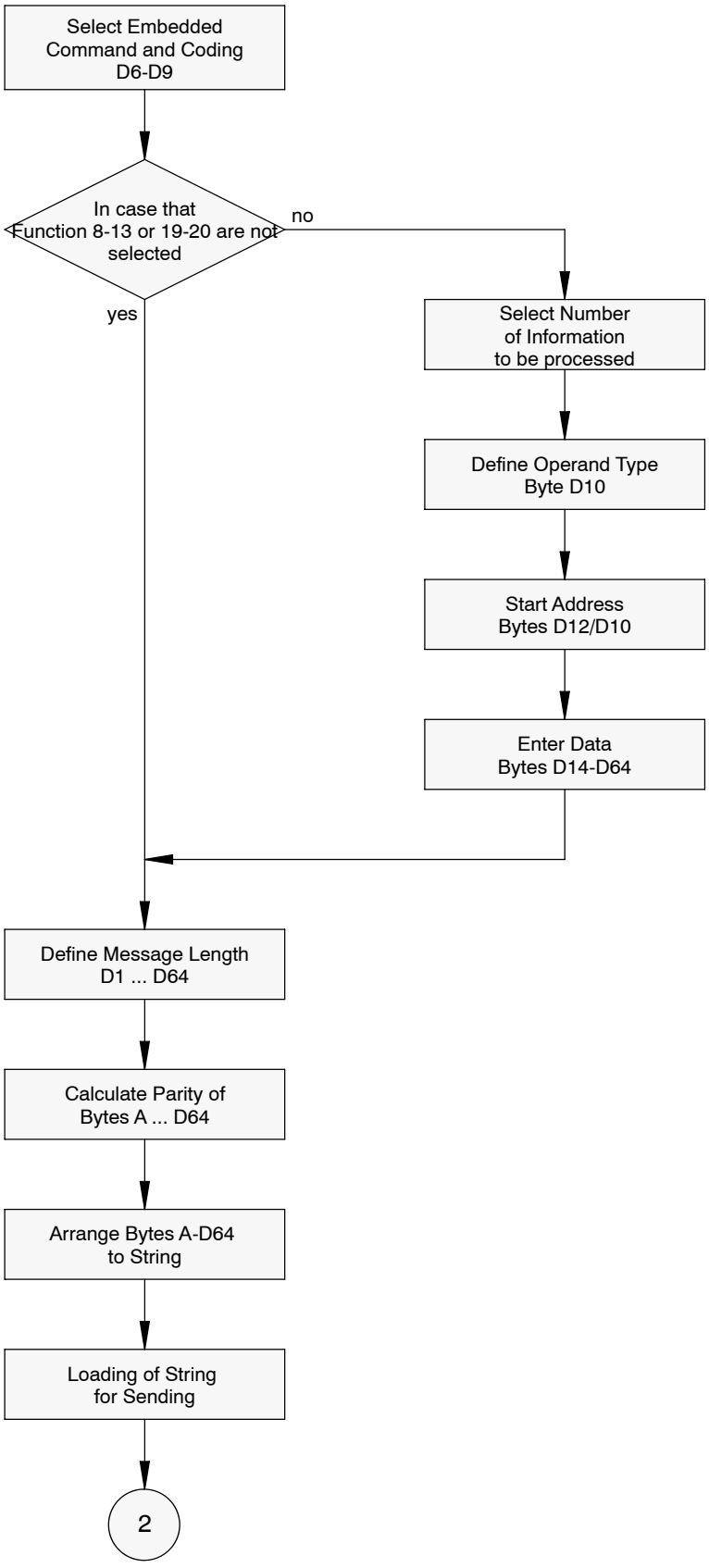


Figure 15 Creation of System Messages

3.3 Analyses of Response Message

For the evaluation of the messages please see chapter 3.2 for details about the linking into the program structure. The most important point for the analysis of the response message is byte D9. In case that byte D9 is different to ”0”, then an error is detected. Dependent on the error type, the developer has to think about the consequences for his program.

In case that no error has been detected, the developer can use the message in the program. The operand type incl. content will be analyzed and processed according to the application requirements, if necessary.

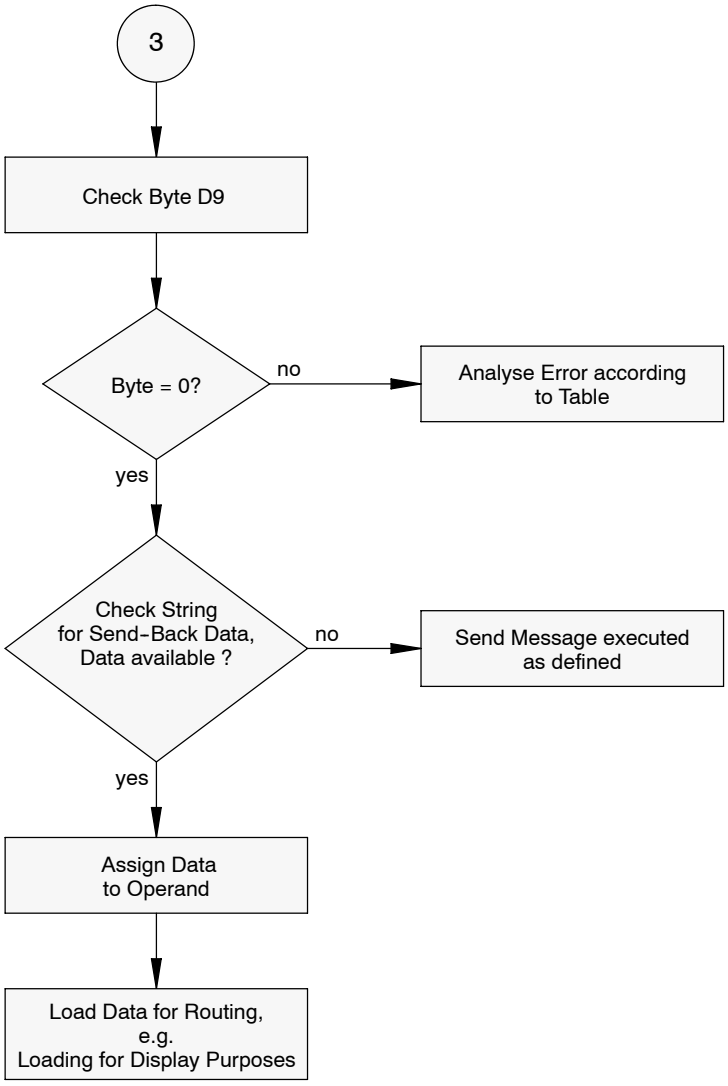


Figure 16 Evaluation of the Response Message

Chapter 4
Examples

This chapter gives some examples about the use of the KS-functions.

4.1 General

The programmable controller (PC) Modicon A120 is able to process one system message (”job message”) per program cycle and is able to load one response into the send buffer, which is then send out. The KS-functions are processed always at the end of the PC-program cycle.

In case that several system messages arrive in short succession, then they are stored in the input buffer.

If, in addition to that, also send conditions are present, then the output messages belonging to it are stored into the send buffer. Therefore it is possible to load several system messages per PC-program cycle. These messages are then send asynchronous to the PC-user program cycle. They are processed according to the polluter-pays principle, i.e. send condition fulfilled or system message arrived. The size of the send buffer is 1 kbyte and can therefore store a larger number of messages before an overflow happens.

The delay for the PC-program cycle is appr. 7 msec. per system message ”single scan”. The declaration of data takes 20 up to 25 msec. The future scan takes less than 7 msec.

When working with send conditions, the capacity of the RS 232 - port can be exceeded. The port works principally with 9600 Baud. When neglecting the timing requirements nearly 900 short messages (length 11 bit) can be transmitted per second. For system messages this number is considerably lower. A complete system message with 55 user bytes has a total length of 69 bytes (69 * 11 bits = 759 bits). Therefore a total number of 12.6 complete system messages can be transmitted per second when neglecting the timing.

When using send conditions this means that the developer has to make sure that at no time more than 10 messages per second should be generated. Otherwise an overflow of the send buffer, which cannot be controlled or checked, might happen. In case of unexpected behavior of the program it is recommended to check the number of generated system messages. A programming routine which generates a system message each PC-program cycle might easily generate up to 100 system messages per second. To be able to prevent this, it is recommended to use a 1/2 Hz flashing pulse in order to control the generation of system messages.

As mentioned before, it is possible that sometimes an overflow of the send buffer occurs even when the KS-functions are used in the correct way. Therefore it might happen, that the master (peripheral) does not receive immediately the related message. Other messages are sent in the meantime.

Therefore it is recommended always to wait for the response message before a new system message is sent. A further possibility is to program a type of sorting mechanism, which allocates afterwards the response messages to the related system messages.

The content of bytes A up to D5 are either already defined or are very easy to determinate (byte F1). The values for the other bytes from D6 up can be taken from the tables in Figure 9 and Figure 10.

The generation of the protection byte is of no importance to the understanding of the KS-function, therefore it is not mentioned in the following examples. The resp. content of the protection byte is marked with ”-”.

For the reason of better understanding all message contents are displayed in hexadecimal format. Please note that the values are shown not complete, i.e. the hex-value ”09” is displayed only as a ”9”.

4.2 Starting and Stopping of the PC-User Program

4.2.1 Starting of the System

The system message send from the peripheral to the Modicon A120 contains the embedded command ”Start User Program according to Hardware- Settings”. The structure of the message is shown in Figure 17. The value for byte F1 is ”9” due to the fact that bytes D1 up to D9 have been used. The bytes D8 and D9 don’t contain necessary data for this KS-function therefore their value is ”0”.

Embedded Command / Bytes	A	F	F1	F2	D1	D2	D3	D4	D5	D6	D7	D8	D9	S
Start User Program (UP) according to HW-Settings	0	0C	9	F0	0	0	1	0	0	1	1	0	0	-

Figure 17 System Message: Start of PC-User Program

In case that no transmission error has been detected, then the slave station Modicon A120 responds with the message as shown in Figure 18. The difference to the request message from the master is byte F in this case, which contains the value ”0D”, i.e. a message from A120 to the master.

Embedded Command / Bytes	A	F	F1	F2	D1	D2	D3	D4	D5	D6	D7	D8	D9	S
Start UP according to HW-Settings	0	0D	9	F0	0	0	1	0	0	1	1	0	0	-

Figure 18 Response Message of the Modicon A120

4.2.2 Stopping of the System

The system message shown in Figure 19 describes the embedded command ”Stop User Program” which is send from the peripheral to the A120.

Embedded Command / Bytes	A	F	F1	F2	D1	D2	D3	D4	D5	D6	D7	D8	D9	S
Stop UP	0	0C	9	F0	0	0	1	0	0	1	4	0	0	-

Figure 19 System Message: Stop PC-User Program

In case that no transmission error has been detected the slave station Modicon A120 responds with the message shown in Figure 20.

Embedded Command / Bytes	A	F	F1	F2	D1	D2	D3	D4	D5	D6	D7	D8	D9	S
Stop User Program	0	0D	9	F0	0	0	1	0	0	1	4	0	0	-

Figure 20 Response Message Modicon A120

4.3 Single Read of three Successive Marker Bits

Please note that the data which are read out of the Modicon A120, should always be a string of successive information and should always be read as a group, e.g. marker bits (Byte D10 = 20 Hex). In the following example three marker bits have to be read out (M31.7, M31.8 and M31.9), i.e. the value of byte D11 is ”3”. These markers are the continuously allocated numbers 999, 1000 and 1001 (999 = 03 E7 Hex.)

The system message of the peripheral to the Modicon A120 contains the embedded command ”Single Read Signal-Memory (SM)-Data”. The message is shown in Figure 21.

Embedded Command / Bytes	A	F	F1	F2	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	S
Single Read SM-Data	0	0C	0D	F0	0	0	1	0	0	5	1	0	0	20	3	03	E7	-

Figure 21 System Message for Single Read of Signal-Memory-Data

In case that no transmission error has been detected the slave station Modicon A120 responds with the message shown in Figure 22. It is assumed that all marker bits have the value ”xx”. Due to the fact that the message is one byte shorter, therefore the byte F1 is now 12 (=0C Hex.) and not 13 (= 0D Hex) any more.

Embedded Command / Bytes	A	F	F1	F2	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	S
Single Read Signal Memory-Data	0	0D	0C	F0	0	0	1	0	0	5	1	0	0	xx	xx	xx	-

Figure 22 Response Message of the Modicon A120

4.4 Multiple Access to declared Data

Besides a single access on data also multiple or cyclical access on data is possible. For this it is necessary to declare these data, i.e. declaration lists have to be defined in the A120. The declared data are then transmitted either event-controlled or on request (cyclical).

4.4.1 Structure of a Declaration List

For multiple read or write tasks on data of the signal memory (SM) it is necessary to define a declaration list in the Modicon A120. This list must be marked unambiguous with a number, the so-called ”declaration number”.

4.4.1.1 How to read a free declaration number

It is always possible to define more than one declaration list. The declaration numbers of the lists are defined in ascending order (starting with the value 17, please compare details on Figure 9). In case that one of the declarations is deleted then a gap in numbering occurs. An additional request might therefore be necessary in order to get the information about the next free declaration number.

The respective system message and response message is shown in Figure 23 and Figure 24. In this example it is assumed that the first ten declaration lists are already defined. The reported free declaration number therefore is 17 + 10 = 27 (=1B Hex.).

Embedded Command / Bytes	A	F	F1	F2	D1	D2	D3	D4	D5	D6	D7	D8	D9	S
Read free Declaration Number	0	0C	9	F0	0	0	1	0	0	6	0	3	0	-

Figure 23 System Message: Read the smallest free Declaration Number

Embedded Command / Bytes	A	F	F1	F2	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	S
Read free Declaration Number	0	0D	0A	F0	0	0	1	0	0	6	0	3	0	1B	-

Figure 24 Response Message of the Modicon A120

4.4.1.2 Declaration of four successive marker bytes for multiple read

In case that four successive marker bytes (e.g. marker bytes 7, 8, 9 and 10) shall be read, then they have to be defined in a declaration list. In this example it is assumed that already ten other declaration lists have been defined before, i.e. the number of the declaration list is ”27” (=1B Hex.).

It is possible to add additional groups of data into the same declaration list, e.g. a group of marker bits or other data. Please be aware that the net data quantity must not exceed 55 bytes. In this example no further data are declared besides the four marker bytes, i.e. no further declaration message is necessary.

Each declaration list must be activated with a so-called ”activating message”. This can be a message ”End Data Declaration” or a message for the declaration of a send condition (for further details please compare chapter 4.4.1.3.

The system message of the peripheral (request) to the A120 contains the embedded command ”Declare SM-Data (multiple read)”. For details please see Figure 25.

Embedded Command / Bytes	A	F	F1	F2	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	S
Declare Signal Memory (SM)-Data (multible read)	0	0C	0D	F0	0	0	1	0	0	2	1	1B	0	21	4	0	7	-

Figure 25 Declaration of four successive marker bytes

In case that no transmission error has been detected then the slave station responds with the message as shown in Figure 26. At this moment no data transmission takes place. It is only used to confirm the correct reception of the declaration message.

Embedded Command / Bytes	A	F	F1	F2	D1	D2	D3	D4	D5	D6	D7	D8	D9	S
Declare SM-Data (multiple read)	0	0D	9	F0	0	0	1	0	0	2	1	1B	0	-

Figure 26 Response Message of the Modicon A120

4.4.1.3 Terminate Data Declaration resp. declare Send Condition

To be able to use a declaration list it always has to be activated. This happens normally by sending the system message ”End Data Declaration”. From this moment the declared data can be read after a request message. The ”activating message” is always related to one declaration list. The declaration number of this list has to be named explicit. Figure 27 shows the corresponding system message where the declaration number ”27” (=1B Hex.) is named.

Embedded Command / Bytes	A	F	F1	F2	D1	D2	D3	D4	D5	D6	D7	D8	D9	S
End Data Declaration	0	0C	9	F0	0	0	1	0	0	1	9	1B	0	-

Figure 27 End Data Declaration

Sometimes it is necessary only to send event-controlled data from the A120 to the master station, e.g. marker bytes. The event in this example is the positive edge of an operand (type ”bit”), i.e. the value of the marker bit M2.1 (or in other format: marker = bit M33 = 21 Hex.) changes from 0 to 1.

This send condition can be defined for each declaration list and terminates the data declaration. The system message (request) therefore has the same effect like the message shown in Figure 29 ”End Data Declaration”. The ”Send Bit Supervision” is then immediately active.

In extension of Figure 8, the structure of the system message (for request of the send condition) is shown in Figure 28.

Caution As already mentioned for Figure 8, the number for the I/O-slots (slots no. 1 up to 18) is shown in byte D9 in decimal format.

In the resp. message the value for D9 has to be entered in hexadecimal format. The byte D8, with its range of values ”RoV”, contains the declaration number for the declaration list, which names the desired marker bytes (for details please see 4.4.1.2). This data declaration must be carried out before the send bit is declared because of the fact that the ”Send Bit Supervision” is active immediately. Please compare the message structure with Figure 8.

	D6	D7	D8	D9	D10	D11	D12	D13
Declare	2	0C	RoV	0	SYS	1	AD	AD
Send Condition	2	0C	RoV	1-18	EA	1	AD	AD

Figure 28 Declare Send Condition

Due to the fact that no input or output bit of an I/O-module is used as send condition, but an internal marker bit (M2.1 = 21 Hex.) of the ALU (byte D9=0) is used as the send bit, therefore byte D8 has the value ”27” (= 1B Hex.), the byte D10 has the value 20 hex. and the bytes D12 and D13 have the values 00 hex. and 21 hex.

The structure of the system message (request) is therefore:

Embedded Command / Bytes	A	F	F1	F2	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	S
Declare Send Condition	0	0C	0D	F0	0	0	1	0	0	2	0C	1B	0	20	1	0	21	-

Figure 29 Declare Send Condition

In case that no transmission error is detected, the A120 responds with the message shown in Figure 31.

Embedded Command / Bytes	A	F	F1	F2	D1	D2	D3	D4	D5	D6	D7	D8	D9	S
Declare Send Condition	0	0D	9	F0	0	0	1	0	0	2	0C	1B	0	-

Figure 30 Response Message of the Modicon A120

4.4.2 Execution of a Declaration List

4.4.2.1 Periodic Scanning of a Declaration List

Loading of the declared data (without send condition) is carried out in principle after the receipt of a corresponding system message (request). This request message can be send cyclical or event-controlled from the peripheral to the A120.

The system message (request) from the peripheral to the A120 contains the embedded command ”Multiple Read Data”. The data which will be transmitted here have been defined in the declaration list with the number 27 (= 1B hex.). The respective message is shown in Figure 31.

Embedded Command / Bytes	A	F	F1	F2	D1	D2	D3	D4	D5	D6	D7	D8	D9	S
Multiple Read Data	0	0C	9	F0	0	0	1	0	0	6	0	1B	0	-

Figure 31 Send Data of the Declaration List 27

With the response message the slave station A120 sends the requested predeclared data. The resp. message is shown in Figure 32.

Embedded Command / Bytes	A	F	F1	F2	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	S
Multiple Read Data	0	0D	0D	F0	0	0	1	0	0	6	0	1B	0	xx	xx	xx	xx	-

Figure 32 Response Message of the Modicon A120

According to the definition of the marker bytes in the declaration list, the marker bytes 7, 8, 9 and 10 are transmitted with data bytes D10 up to D13. For further details please see chapter 4.4.1.2.

4.4.2.2 Polling while waiting for the Send Bit

In case that a send condition has been declared then the short message SMG 80 is used. When using the short message SMG 81 the programmable controller A120 does not send any system messages.

4.4.3 Clearing of Declarations

The declarations cannot be cleared (deleted) in the A120. This can be done only when the program is loaded again into the programmable controller (PC). When the user program is loaded into the PC then all memory areas are standardized, i.e. in this case all declaration list must be defined again.

Using the on-line-exchange function of Dolog AKF12 has no influence on the declaration lists.

It is possible to generate an overflow when too many declaration lists are send to the A120. In the case of doubt it is recommended to clear all declaration lists and to generate new lists. It is also possible to clear only parts of these list. In difficult situations the next free declaration number can be found as mentioned in chapter 4.5.

When all declaration lists are cleared (deleted) also other lists are lost. Is for example Viewstar 200 XA used in the application, then also the lists for this application are cleared.

The resp. messages for clearing are shown in Figure 33 and Figure 34. The response messages are not mentioned again and are structured as already explained before in other examples. In Figure 34 please see how a single list with the declaration number 1B hex. (= 27 in decimal format) is cleared. In order to be able to delete all declaration numbers the value ”0” is entered as the declaration number.

Embedded Command / Bytes	A	F	F1	F2	D1	D2	D3	D4	D5	D6	D7	D8	D9	S
Clear Declaration	0	0C	9	F0	0	0	1	0	0	1	7	1B	0	-

Figure 33 Cleanse single Declaration List in A120

Embedded Command / Bytes	A	F	F1	F2	D1	D2	D3	D4	D5	D6	D7	D8	D9	S
Clear Declaration	0	0C	9	F0	0	0	1	0	0	1	7	0	0	-

Figure 34 Clear all Declaration Lists in A120

4.5 Using Timers and Counters

For reading or writing purposes of timers and counters the A120 offers two possibilities. The older one, upward compatible solution, which is available for all firmware versions has to be used with care. When working with this older version the actual and setpoint values as well as the status of the timer or counter are overwritten simultaneously.

This version won't be subject of this documentation.

The newer solution, which is available in firmware versions

On the other hand, the new solution, which is however available only from firmware level 3, allows an individual access to the actual as well as setpoint value.

The illustrated message shows the job telegram with which at a definite timer (Byte D11 = 1) with the number 16 (Bytes D12+D13 = 0010 Hex) the setpoint value (Byte D10 = 3D Hex) is set to 33 (Byte D14+D15 = 0021 Hex).

Embedded command / Bytes	A	F	F1	F2	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D16	S
Single write SM data	0	0C	10	F0	0	0	1	0	0	8	0	8	0	13	5A	0C	0C	0C	0	0	-

Figure 35 Set timer setpoint value of timer 16 to 33

The response message looks as follows:

Embedded command / Bytes	A	F	F1	F2	D1	D2	D3	D4	D5	D6	D7	D8	D9	S
Single write SM data	0	0C	9	F0	0	0	1	0	0	7	1	0	0	-

Figure 36 Response message of A120

4.6 Changing the Time and Date

Time and date can be changed with the message shown below. For example to change over from summer to winter time. The date and time readings are achieved basically via the corresponding system marker.

In the following example the time and date 12 a.m. on 12.12.1990 should be set. The corresponding job message is illustrated in Figure 37. The corresponding response message is not shown here again.

Embedded command / Bytes	A	F	F1	F2	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D16	S
Single write SM data	0	0C	10	F0	0	0	1	0	0	8	0	8	0	13	5A	0C	0C	0C	0	0	-

Figure 37 Write time and date

Here:

D10 = centuries
D11 = year
D12 = month
D13 = day
D14 = hour
D15 = minute
D16 = second

4.7 Reading of the System Declaration List

The system declaration list contains data about the number of the configured markers, system markers, timers and counters. This list is already defined, declared in the A120 CPU and has the declaration number 1 constantly assigned to it. It will be read out with the message type ”multiple read” as described in chapter 4.4.1.

The data structure of the response message is shown in the following. The data has the format ”Word”.

D10/D11	=	No. of marker-bits
D12/D13	=	No. of marker-bytes
D14/D15	=	No. of marker words
D16/D17	=	No. of marker double words
D18/D19	=	No. of system marker bits
D20/D21	=	No. of system marker bytes
D22/D23	=	No. of system marker words
D24/D25	=	No. of system marker double words
D26/D27	=	No. of timers
D28/D29	=	No. of counters
D30 to D54	=	other parameters

4.8 Reading of the system status list

The system status list contains data about the type of CPU, the DIP switch setting and the firmware version. This list is already defined, declared in the A120 CPU and has the declaration number 2 permanently assigned to it. It will be read out with the message type ”multiple read” as described in chapter 4.4.2.

The data structure of the response message is shown below. The data have the format ”byte”.

D10	=	Identcode of the CPU
D11	=	DIP switch position of the 8 pole switch
D12	=	DIP switch position of the 4 pole switch
D13	=	ab
D14	=	cd
D15	=	ef
D16	=	gh
D17	=	ik
D18 to D63	=	other parameters

From this the part number follows as : abcdefgh.ik

For exampel: ab = 00 dec, cd = 27 dec, ef = 66 dec, gh = 03 dec, ik = 03 dec

It follows the part number is _276603.03.